Differentiating Congestion vs. Random Loss: A Method for Improving TCP Performance over Wireless Links

Christina Parsa J.J. Garcia-Luna-Aceves

Computer Engineering Department
Baskin School of Engineering
University of California
Santa Cruz, California 95064
chris, jj@cse.ucsc.edu

Abstract—**Recent research has focussed on the problems associated with TCP performance in the presence of wireless links and ways to improve its performance. We present an extension to TCP Santa Cruz which improves TCP performance over lossy wireless links. TCP has no mechanism to differentiate random losses on the wireless link from congestion, and therefore treats all losses as congestive. We present a simple method in which our protocol is able to differentiate these random losses, thereby avoiding the rate-halving approach taken by standard TCP whenever any loss is detected. We compare the performance of our protocol against TCP Reno and demonstrate higher throughput and lower end-to-end delay with our approach.**

I. Introduction

The use of wireless information devices to access the Internet and the WWW, in particular, is an ever-increasing practice of mobile users world-wide, resulting in the need for reliable client-server communication over wireless links. Unfortunately, the de-facto Internet protocol for reliability, TCP, has severe performance problems when operated over wireless links [12][2][11]. Recent research has focussed on the problems associated with TCP performance in the presence of wireless links and ways to improve its performance. The key issue lies at the very heart of TCP's congestion control algorithms: namely, packet loss is the only detection mechanism for congestion in the network. Wireless links are inherently lossy and in addition to random losses, they suffer from long periods of fading as well. TCP has no mechanism to differentiate these losses from congestion, and therefore treats all losses as congestive by reducing its transmission window (and in effect halving the throughput of the connection). Many proposals to improve TCP performance have focussed on hiding wireless losses from TCP by performing retransmissions of any lost data before TCP notices the loss [1][3][6][4][5][11]. There is far less research on methods for TCP to differentiate between losses due to congestion and those due to noise on a wireless channel [7] [9]. In this paper we extend our protocol TCP Santa Cruz [10] to identify losses due to congestion from those caused by a lossy wireless link. Because it cannot be assumed that a reliable link layer or some other method of error recovery exists at the wireless interface, methods of discovering random wireless losses at the TCP source must be in place. Once losses are identified as random, there is no need to reduce TCP's transmission rate because the losses are not due to congestion. TCP Santa Cruz monitors the queue developing over a bottleneck link and thus determines whether congestion is increasing in the network; it can then identify losses as either congestive or random and respond appropriately. Another proposed method to identify wireless losses at the source [9] uses variation in RTT measurements as an indication of congestion in the network. Our method does not use RTT and as a result is better suited for this application because RTT measurements cannot differentiate between an increase due to congestion on the forward path or on the reverse path.

This work was supported in part at UCSC by the Office of Naval Research (ONR) under Grant N00014-99-1-0167.

II. TCP Santa Cruz

TCP Santa Cruz [10] is a new implementation of TCP that detects not only the initial stages of congestion in the network, but also identifies the direction of congestion, i.e., it determines whether congestion is developing in the forward or reverse path of the connection. TCP Santa Cruz is then able to isolate the forward throughput from events such as congestion that may occur on the reverse path. Congestion is determined by calculating the relative delay that one packet experiences with respect to another as it traverses the network; this relative delay is the foundation of our congestion control algorithm. The relative delay is used to estimate the number of packets residing in the bottleneck queue; the congestion control algorithm keeps the number of packets in the bottleneck queue at a minimum level by adjusting the TCP source's congestion window. The congestion window is reduced if the bot-

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2000		**00-00-2000 to 00-00-2000**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Differentiating Congestion vs. Random Loss: A Method for Improving TCP Performance over Wireless Links	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**4**	

tleneck queue length increases (in response to increasing congestion in the network) beyond a desired number of packets (n). The window is increased when the source detects additional bandwidth availability in the network (*i.e.*, after a decrease in the bottleneck queue length). TCP Santa Cruz can be implemented as a TCP option by utilizing the extra 40 bytes available in the options field of the TCP header. A complete description of the protocol is given in previous work [10].

III. Applying TCP Santa Cruz to wireless links

As congestion develops in a wired network, data packets fill the network links. If the rate of data entering the network continues to increase, data packets fill the network queues until the queues reach capacity and the routers begin to drop packets. In other words, losses due to congestion are preceded by an increase in the network bottleneck queue. The basic methods used to detect congestion in TCP Santa Cruz are easily applied to networks containing wireless links and are ideal for distinguishing congestion versus random loss in a network which may or may not contain a wireless link.

TCP Santa Cruz easily identifies a congestive loss as one which is preceded by an increase in the bottleneck queue length. A wireless loss, on the other hand, can be identified as a random loss that is **not** preceded by a buildup in the bottleneck queue. TCP Santa Cruz monitors changes in the bottleneck queue over an interval equal to the amount of time it takes to transmit one window of data and receive acknowledgments corresponding to all the packets transmitted in the window. Figure 1 shows how the protocol counts intervals of queue buildup. The protocol starts in state *count = 0* which means we have not noticed an interval in which the bottleneck queue increased. Once an interval experiences an increase in queue length, we transition to the state *count = 1*. An interval with a decrease or steady queue size causes a transition back. Finally, if a loss occurs in state *count = 2*, we conclude that the loss was due to congestion. At that point the congestion avoidance algorithm is followed and the sender's transmission window is reduced in half. However, if a loss is not preceded by at least two consecutive intervals of increasing queue length, we infer that it is a random loss and the congestion avoidance algorithm is not followed and the transmission window is maintained at its current size. We have chosen the constraint of two intervals of increasing queue length (instead of just one) as the signal for congestion in order to avoid any noise in the network. It remains an area of future work to evaluate this value. TCP Santa Cruz reduces the transmission rate only when congestion is identified as the cause of lost packets, otherwise, wireless losses can simply be quickly retransmitted without a reduction in the data transmission rate.

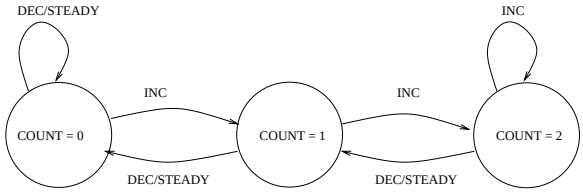


Fig. 1. State diagram for monitoring buildup of network queue interval

IV. Performance Results

We have implemented TCP Santa Cruz in the NS network simulator [8] and compared its performance to TCP Reno over the network shown in Figure 2. The network consists of a wired network segment connected to a base station with an interface to a wireless LAN. Losses are randomly applied to the wireless link with an exponential distribution. The bandwidth delay product (BWDP) of this network is 480 Kbits (6 1Kbyte packets) and the queue at the base station is set to accommodate up to 12 1Kbyte packets.

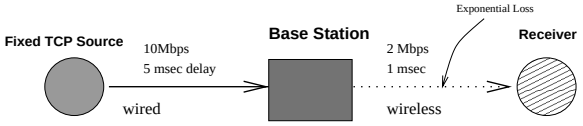


Fig. 2. Network used for simulations.

A. Random losses without network congestion

In this experiment we perform FTP transfers from the fixed host to the wireless receiver and examine the operation of TCP Santa Cruz compared to TCP Reno for a low wireless bit error rate of $1 \cdot 10^{-6}$. At this rate, on the average, 1 out of every 125 packets are dropped on the wireless link for both the Santa Cruz and Reno transfers. Each simulation is run for 60 seconds.

Figure 1 showed how TCP Santa Cruz counts the consecutive intervals over which it notices a bottleneck queue increases in order to determine if congestion is present when a loss is detected. Figure 3 shows the $count$ value for this state diagram as the FTP transfer progresses. Since the value of $count$ is rarely equal to two, we would expect nearly all losses on the wireless link to be considered as random by the protocol; in other words, once the losses are discovered, we expect the protocol to simply retransmit most losses without reducing the transmission window.

Figure 4(a) shows the congestion window for TCP Santa Cruz. We notice that the congestion window never drops to half of its value when these losses occur on the wireless link, verifying that, in this simulation, TCP Santa Cruz correctly identifies all losses as random losses and not due to congestion. There is one instance, however, around time $t = 16$ when there is a TCP timeout. This is

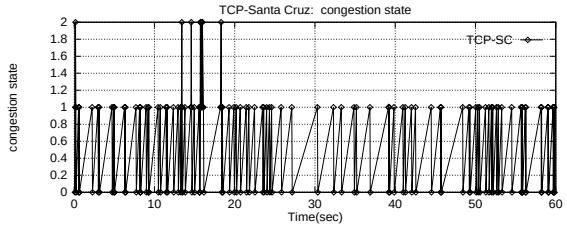


Fig. 3. Congestion state of TCP Santa Cruz

because a retransmitted packet is once again dropped on the wireless link.

This simulation is run with $n = 4$, which means that the TCP Santa Cruz will try to fill the bit pipe and maintain an extra 4 packets in the bottleneck queue (at the base station). The graph shows that TCP Santa Cruz indeed maintains its window around 10 packets, exactly the BWDP (6 packets) plus the extra 4 packets in the queue. This steady value of the congestion window is directly correlated to the number of packets in the bottleneck queue. In Figure 4(b), on the other hand, we show the congestion window for TCP Reno, which oscillates wildly between 1 and 25 packets (and at times up to 60 packets during the Fast Recovery phase). The variation in the congestion window is directly responsible for the long end-to-end delays experienced by transfers using TCP Reno.

Figures 5(a) and (b) show the delay experienced by each packet through the network. We see that the delay experienced by packets in TCP Santa Cruz is roughly half the delay with TCP Reno. In addition, the variance in delay is an order of magnitude smaller. Because Reno keeps so many packets in the bottleneck queue, each packet experiences a considerable amount of queueing before it is transmitted over the wireless link. The only time packets experience low delay is immediately following either a timeout (when the congestion window is reduced to one segment) or after a fast retransmit (when the congestion window is cut in half). Shortly thereafter the queue again builds and the delay once again increases.

With respect to throughput it is important to mention that for this error rate TCP Reno does not experience an appreciable reduction in throughput because it is able to recover most errors via the fast retransmission mechanism and therefore is able to keep its window large enough to fill the bit pipe of the connection.

B. Various error rates

Next we perform simulations for a variety of wireless link error rates. We demonstrate that because our algorithm identifies random losses on the wireless link, it outperforms Reno both in terms of throughput and end-to-end delay. Figure 6(a) shows the throughput achieved by Santa Cruz and Reno in the presence of increasing error rates on the wireless link. The wireless error rate is increased from $1 \cdot 10^{-6}$ (corresponding to 1 packet loss per 125 packets) to $1 \cdot 10^{-5}$ (corresponding to 1 packet loss per 12.5 packets). TCP Santa Cruz provides higher throughput than Reno in all cases. In fact, the throughput of Santa Cruz does not drop significantly until very high error rates (1 packet per 12.5). This drop is due to the fact that so many of the retransmissions are also subsequently dropped. The current code in this case defaults to a timeout for the second retransmission, but it would be possible to check on the status of the retransmitted packet whenever an ACK arrives because Santa Cruz keeps a timestamp for every transmission. This would improve performance for these catastrophic error rates.

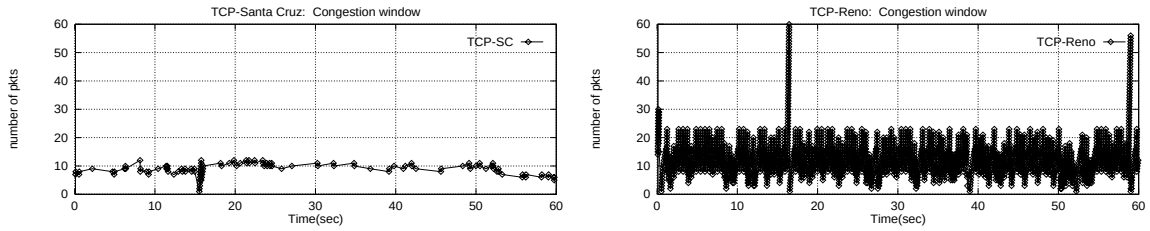
Figure 6(b) shows how the end-to-end delay changes as the error rates increase. The reason Reno's delay decreases for the first three points is that as more time is spent in timeouts, the bottleneck queue decreases and subsequent packet experience a smaller delay. At the rate 1/12.5 packet loss Reno experiences a very large delay variance because packets either make it through quickly with a small bottleneck queue, or they experience a long delay due to timeouts. The delay in Santa Cruz is fairly steady until the large error rate. At this time so many dropped packets are also dropped on the retransmission that timeouts are experienced as well.

V. CONCLUSION

We have shown through simulation that TCP Santa Cruz is able to maintain high throughput over the wireless link for a range of error rates because it does not reduce its sending rate when the losses are determined to be random losses on the wireless link. TCP Reno, on the other hand, can make no such distinction and as a result shows reduced performance, even when link error rates are low. Packets transmitted with TCP Santa Cruz also experience a lower end-to-end delay and delay variation from source to receiver because the protocol keeps the bottleneck queue at a minimum level and does not create the oscillations in bottleneck queue length that is typical of the TCP Reno transmission pattern.

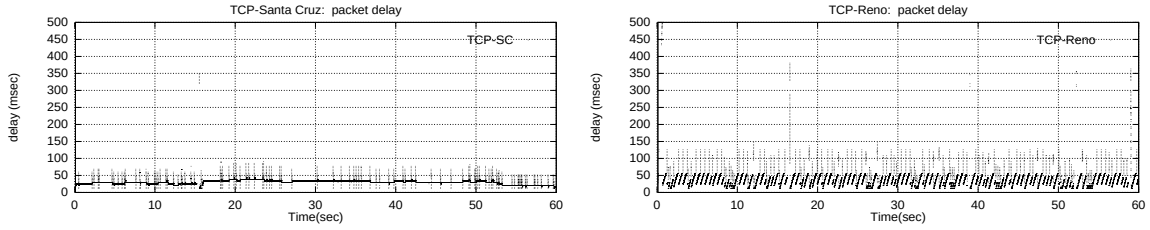
The advantage of our approach is that we are able to correct for losses on a wireless link at the source without relying on help from the link layer (either in the form of a reliable link layer or FEC) or a proxy agent at a base station. The method of using relative delays to determine congestion and to monitor the increasing or decreasing congestion in the network is well-suited to this problem. The alternative, RTT monitoring, cannot take into account the effects of congestion on the reverse path as a contributing factor to increased RTT measurements.

Our future work will focus on evaluating the performance of our approach when congestion and random wireless losses occur simultaneously. The difficulty lies in identifying random losses that occur during periods of peak congestion.



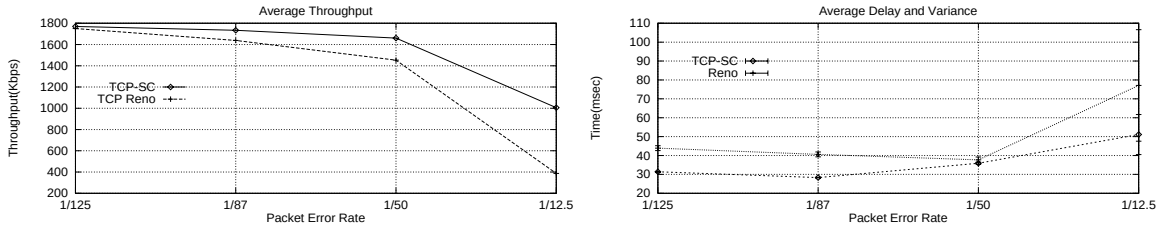
(a) (b)

Fig. 4. Congestion window, BER of 10^{-6} (a)TCP-Santa Cruz (b) TCP Reno



(a) (b)

Fig. 5. End-to-end delay per packet, BER of 10^{-6} (a) TCP-Santa Cruz (b) TCP Reno



(a) (b)

Fig. 6. TCP Santa Cruz and Reno for different error rates (a) Throughput (b) Delay and variance

REFERENCES

[1] A. Bakre and B. R. Badrinath. I-TCP: indirect TCP for mobile hosts. In *Proc. 15th IEEE Int'l Conf. on Distributed Computing Systems*, pages 136–43, Vancouver, BC, Canada, May 1995.

[2] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 6(5):756–69, Dec. 1997.

[3] H. Balakrishnan, S. Seshan, and R. Katz. Improving reliable transport and handoff performance in cellular wireless networks. *ACM Wireless Networks*, Dec. 1995.

[4] Kevin Brown and Suresh Singh. M-TCP: TCP for mobile cellular networks. In *Computer Communication Review*, volume 27 No. 5, pages 19 – 43, Oct., 1997.

[5] R. Caceres and L. Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal on Selected Areas in Communications*, 13(5), 1995.

[6] H. Chaskar, T.V. Lakshman, and U. Madhow. On the design of interfaces for TCP/IP over wireless. In *MILCOM '96 Conference Proceedings*, volume 1 No. 3, pages 199–203, Oct. 1996.

[7] J. Cobb and P. Agrawal. Congestion or corruption? a strategy for efficient wireless TCP sessions. In *Proceedings IEEE Symposium on Computers and Communications*, June 1995.

[8] S. McCanne and S. Floyd. Ns network simulator. http://www-nrg.ee.lbl.gov/ns/.

[9] N.K.G.Samaraweera. Non-congestion packet loss detection for TCP error recovery using wireless links. In *IEE Proceedings-Communications*, volume 146, pages 222–30, Aug. 1999.

[10] C. Parsa and J.J. Garcia-Luna-Aceves. Improving TCP congestion control over internets with heterogeneous transmission media. In *1999 International Conference on Network Protocols*, pages 213–21. IEEE, Oct. 1999.

[11] C. Parsa and J.J. Garcia-Luna-Aceves. Improving TCP performance over wireless networks at the link layer. *ACM Mobile Networks and Applications Journal*, 5(1):57–71, 2000.

[12] J.B. Postel. Transmission Control Protocol. Technical report, SRI Network Information Center, September 1981. RFC 793.